RECURSION THEORY
LECTURE IV

16 NOVEMBER 2021

MODELS OF COMPUTATION — Defining M-computability

$$M = (\Sigma, \Phi, k)$$

ADDITIONAL PROPERTIES

✓ (1) COMPOSITIONALITY — if f, g computable, then $f \circ g$.

✓ (2) CASE DISTINCTION — if $x, y \in \Sigma^* \cup \{\uparrow\}$

$$d_{xy} : v \longmapsto \begin{cases} x & \text{if } v = \varepsilon \\ y & \text{if } v \neq \varepsilon \end{cases} \quad \text{computable}$$

✓ (3) IDENTITY — $id : \Sigma^* \longrightarrow \Sigma^*$ computable

✓ (4) UNIVERSALITY

Splitting function: $w \longmapsto w_E$, $w \longmapsto w_O$ are computable

$$w \longmapsto f_{w_E}(w_O) \quad \text{is computable}$$

✓ (5) DUPLICATION

$$\sigma_0\sigma_1 \ldots \sigma_u = w \longmapsto \sigma_0\sigma_0\sigma_1\sigma_1 \ldots \sigma_u\sigma_u$$

is computable

✓ (6) DOMAIN CHECK — if f, g computable, then

$$c^{dom}_{fg} : v \longmapsto \begin{cases} g(v) & v \in dom(f) \\ \uparrow & v \notin dom(f) \end{cases}$$

computable.

X (7) RANGE CHECK — if f, g computable, then

$$c^{ran}_{fg} : v \longmapsto \begin{cases} g(v) & v \in ran(f) \\ \uparrow & v \notin ran(f) \end{cases}$$

↑ TO BE DONE

Informal check of "reasonable-ness".

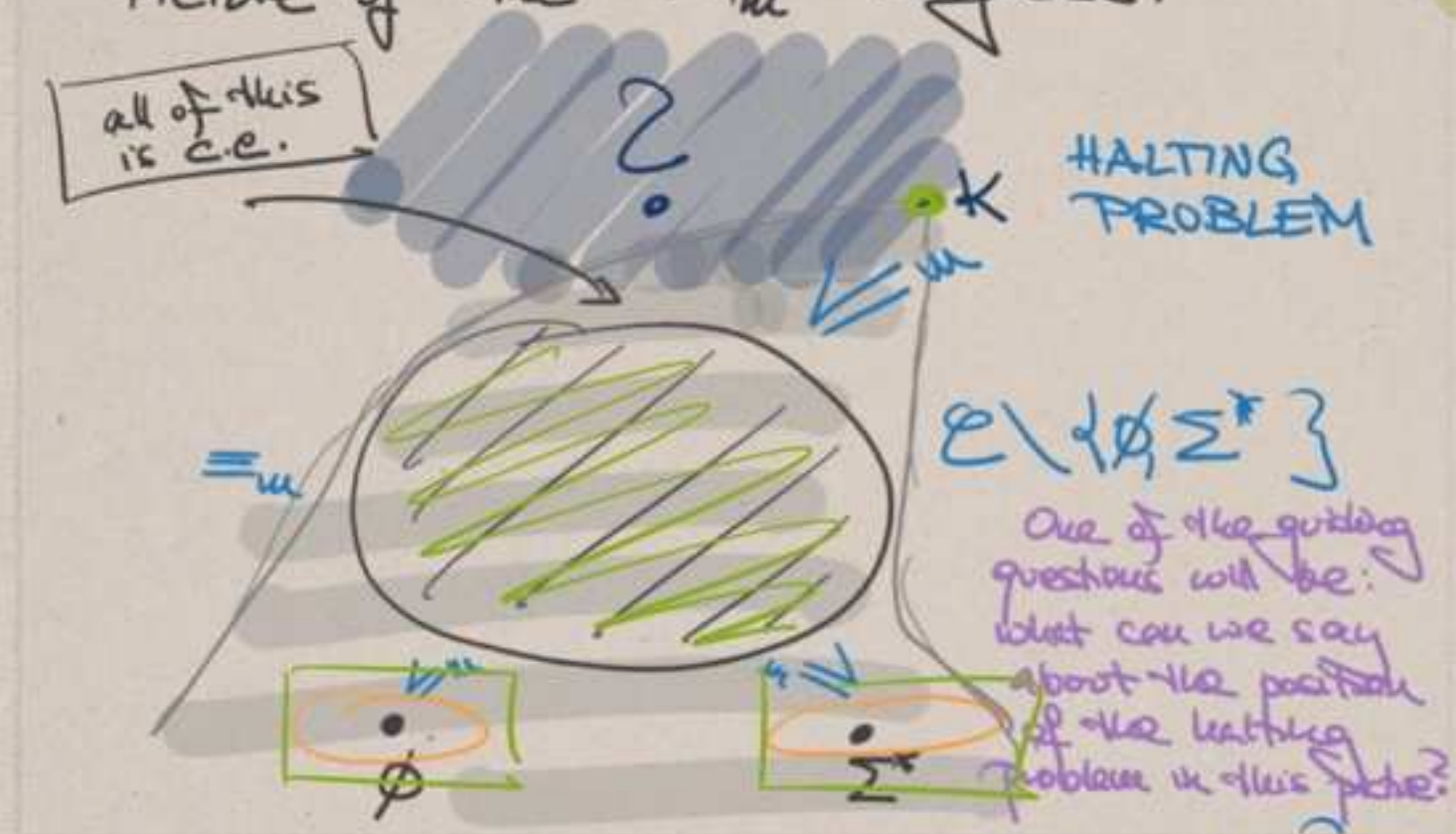
If $\mathbb{M}$ is a model of computation with properties ① to ④, then

- if $A \leq_m B$ and B is computable or computably enumerable, then so is A.
- The HALTING PROBLEM K is computably enumerable, but not computable.

FROM LECTURE III:

MANY-ONE-DEGREES

Picture of the $\equiv_m$-degrees:



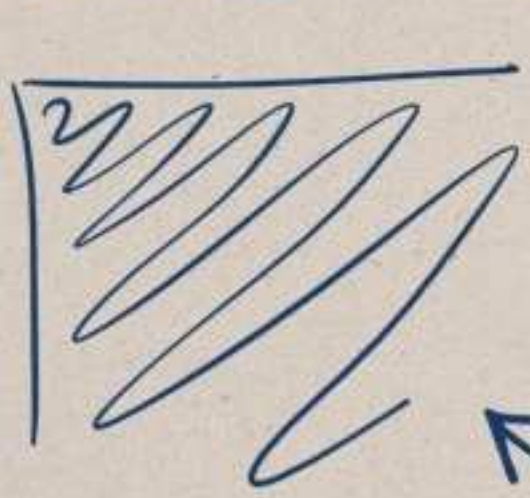
RANGE CHECK If f, g are computable, then

So is

$$C^{ran}_{f,g} : w \longmapsto \begin{cases} g(w) & \text{if } w \in ran(f) \\ \uparrow & \text{if } w \notin ran(f) \end{cases}$$

INFORMAL ARGUMENT THAT WE EXPECT "RANGE CHECK" TO BE TRUE FOR A REASONABLE MODEL OF COMPUTATION.

There is an intuitively computable bijection between $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$:



When we prove that $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$ are in bijection (in first year B.Sc. classes), we usually draw this picture

We can write this down as:

$$\langle i, j \rangle \longmapsto \frac{(i+j)(i+j+1)}{2} + j.$$

$$\langle i,j \rangle \longmapsto \frac{(i+j)(i+j+1)}{2} + j$$

$$0,0 \longmapsto 0$$

$$0,1 \longmapsto \frac{(0+1)(0+1+1)}{2} + 1 = 2$$

$$1,0 \longmapsto \frac{(1+0)(1+0+1)}{2} + 0 = 1$$

$$2,0 \longmapsto \frac{(2+0)(2+0+1)}{2} + 0 = 3$$

$$1,1 \longmapsto \frac{(1+1)(1+1+1)}{2} + 1 = 4$$

	0	1	2	3	4	5	6	7
0	0	1	3	6	10			
1	2	4	7	11				
2	5	8	12					
3	9	13						
4	14							
5								
6								

The concrete representation as a formula only using operations that we demand to be computable [multipl. & addition] shows that we can expect $i,j \longmapsto \langle i,j \rangle$ to be computable.

There is a simple algorithm to list all words in Σ^* such that the i-th word in the list can be produced by a computer program: $\Sigma = \{\sigma_0, \dots, \sigma_n\}$
$|\Sigma| = n+1$
Since Σ is a finite alphabet, we can just list:

① First the empty word $\varepsilon =: w_0$.

② Then all words of length one:

$$w_1 := \sigma_0$$
$$w_2 := \sigma_1$$
$$\vdots$$
$$w_{n+1} := \sigma_n$$

③ Then all words of length two in lexicographic order:

$(n+1)^2$ {
$\sigma_0\sigma_0, \sigma_0\sigma_1, \sigma_0\sigma_2, \dots, \sigma_0\sigma_n,$
$\sigma_1\sigma_0, \sigma_1\sigma_1, \dots \sigma_1\sigma_n,$
$\vdots$
$\sigma_n\sigma_0 \dots \sigma_n\sigma_n$
}

④ and so on.

This can be done by a computer program that gets k as input and produces w_k as output.

OBSERVE that the following algorithm doesn't work:

> Take the natural numbers in turn, check w_i in the i-th step of the algorithm, if $w = f(w_i)$, then output $g(w)$.

[Reason: if, e.g., $\varepsilon = w_0 \notin \mathrm{dom}(f)$, then this algorithm will never get beyond the 0-th step]

INSTEAD

(*) Fix k. Describe the k-th step:
Find i and j s.t. $\langle i,j \rangle = k$.
Take w_i [computable $i \longmapsto w_i$]
Run the f-computation on w_i for j steps!
If it result halted, move on to step $k+1$.

(*) If it has halted, we obtained $f(w_i)$.
Check whether $f(w_i) = w$.
If it isn't, move on to step $k+1$.
If it is, $w \in \mathrm{ran}(f)$.
Now apply the g-computation to w and output $g(w)$.

We claim that (*) computes g:

Fix w. Case 1 $w \in \mathrm{ran}(f)$.
This means that there is some word w_i s.t. $w = f(w_i)$.
And some j s.t. the $f(w_i)$-computation halts after j steps.
Therefore, let $k := \langle i, j \rangle$.
Thus (*) will produce $g(w)$ in the k-th step of the procedure [at the latest].

And the k-step is reached since every substep of (*) takes a finite number of computation steps.

Case 2 $w \notin \mathrm{ran}(f)$

In that case, the check $f(w_i) = w$ will never succeed, and thus none of the substeps of (*) halts.

Therefore (*) produces $\uparrow$.

Together (*) produces $C_{f,g}^{\mathrm{ran}}$.

REMARK We are performing ∞ many computations "in parallel" [more precisely in a potentially infinite series]. This works because $C_{f,g}^{\mathrm{ran}}$ demands $\uparrow$ precisely when the check fails.

Summary

We now know what a "reasonable" model of computation is and how it behaves.

BUT: we don't know yet that they exist!

We're now going to see three standard models of computation:

TURING MACHINES

230 A. M. TURING. [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

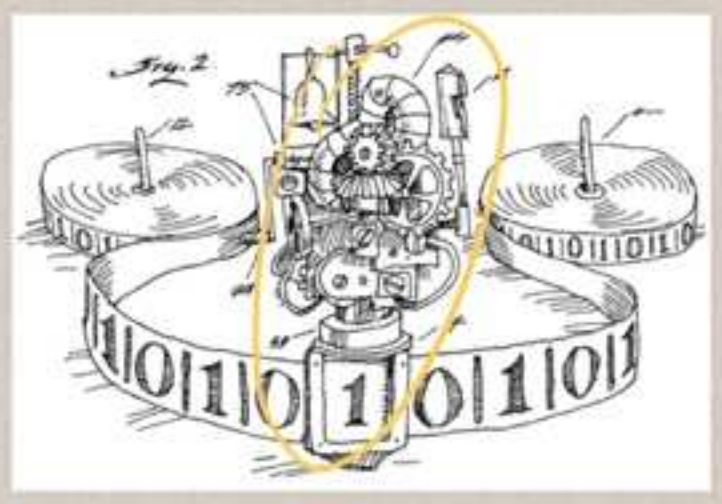
[Received 28 May, 1936.—Read 12 November, 1936.]

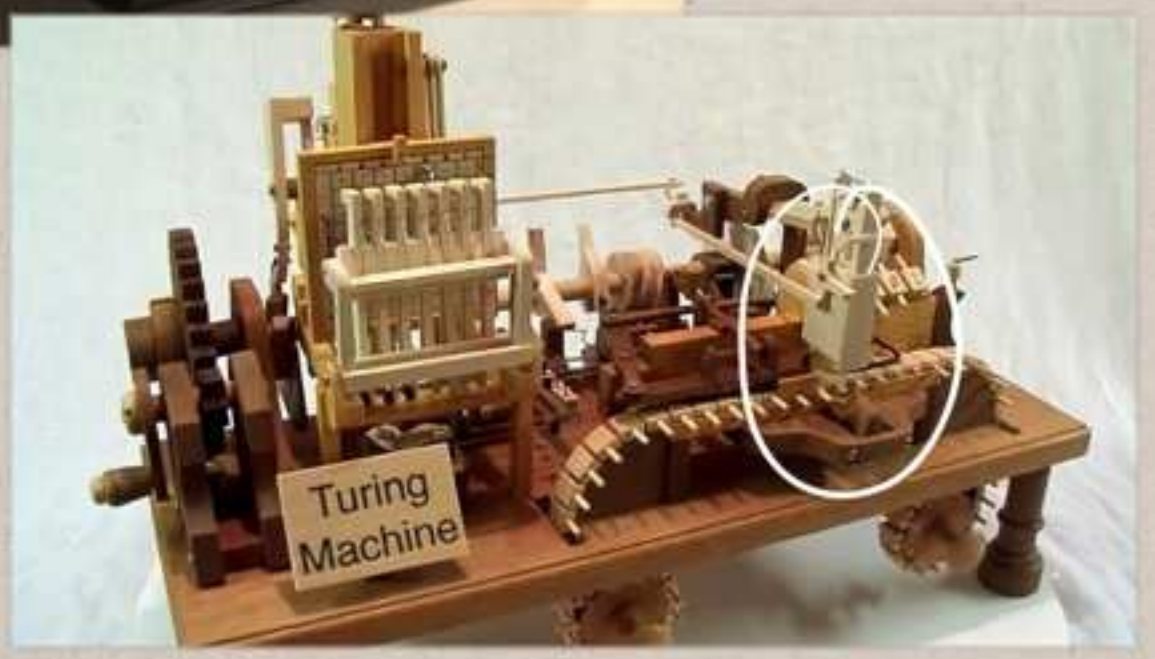
The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable numbers, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte Math. Phys.*, 38 (1931), 173–198.





Def. A Turing machine consists of an (one-side) infinite tape [cells indexed with natural numbers] and a read/write head that sees one cell at a time and operates based on what it sees.

Set of states S; $s_0, s_1 \in S$ finite
(s_0: START, s_1: HALT)

At each moment, the situation of the machine is described by a tuple from $\mathbb{N} \times S \times \Sigma^*$ — CONFIGURATIONS

($\mathbb{N}$: R/W head position; S: current state; Σ^*: tape content)

A TM program is a function

$$S \times \Sigma \cup \{\emptyset\} \longrightarrow S \times \Sigma \cup \{\emptyset\} \times \{\leftarrow, \rightarrow, \emptyset\}$$

($\emptyset$: new symbol for "no symbol")

Observe that for a fixed S and fixed Σ, a TM program is a finite function.

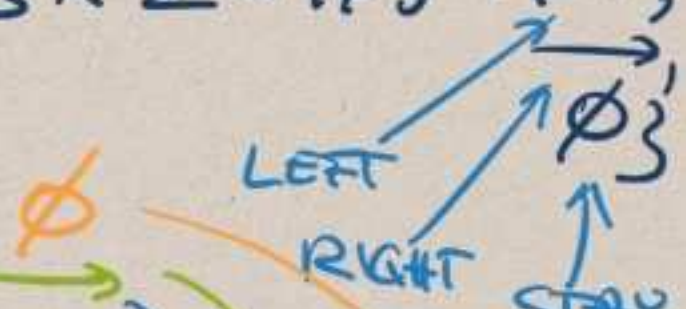
Thus, if Σ has more than one symbol, there is some (computable) way to encode TM programs as words in Σ^*.

$$S \times \Sigma \cup \{\emptyset\} \longrightarrow S \times \Sigma \cup \{\emptyset\} \times \{\leftarrow, \rightarrow, \emptyset\}$$

$$(s, \sigma) \longmapsto (s', \sigma', \leftarrow)$$

is read as

IF YOU'RE IN STATE s
AND READ σ, WRITE σ',
GO TO STATE s' AND
MOVE THE HEAD ONE CELL
LEFT
RIGHT
DO NOT MOVE THE HEAD



INTERPRET TURING MACHINES AS MODELS OF COMPUTATION.

We have to think of the input as a configuration rather than just as the tape content.

$$\overline{\Sigma} := \Sigma \cup \{\emptyset\} \cup \{\sigma_s \mid s \in S\}$$

STATE SYMBOLS representing the states of the machine.

Then (some) words in $\overline{\Sigma}^*$ describe configurations:

Suppose there is a unique state symbol in the word $w \in \overline{\Sigma}^*$:

$$w = \sigma_0 \sigma_1 \dots \sigma_{n-1} \boxed{\sigma_s} \sigma_n \dots \sigma_m$$

ONLY STATE SYMBOL

Interpreted as: $\sigma_0 \sigma_1 \dots \sigma_{n-1} \sigma_n \dots \sigma_m$ is the tape content; the head is at position n; the head is in state s.

So now a TM program P yields a transition function Φ as in our definition of models of computation:

Let $w \in \Sigma^*$ be a configuration, so

$$w = \sigma_0 \cdots \sigma_{n-1} \boxed{\sigma_s \; \sigma_n} \cdots \sigma_m$$

Describe $\Phi(P, w)$:

if $P(s, \sigma_n) = (s', \sigma', \leftarrow)$, then

$$\sigma_0 \cdots \sigma_{n-2} \boxed{\sigma_{s'} \sigma_{n-1} \sigma'} \sigma_{n+1} \cdots \sigma_m$$

if $P(s, \sigma_n) = (s', \sigma', \rightarrow)$, then

$$\sigma_0 \cdots \sigma_{n-1} \boxed{\sigma' \sigma_{s'} \sigma_{n+1}} \cdots \sigma_m$$

if $P(s, \sigma_n) = (s', \sigma', \emptyset)$, then

$$\sigma_0 \cdots \sigma_{n-1} \boxed{\sigma_{s'} \sigma'} \sigma_{n+1} \cdots \sigma_m$$

The halting bit of $\Phi(P, w)$ is 1 iff $s' = s_1$.

REGISTER MACHINES

1961:

various people develop the idea that later becomes RM

von Neumann architecture

a single storage unit for everything

Harvard architecture

separate working memory & storage

Harvard Mark I

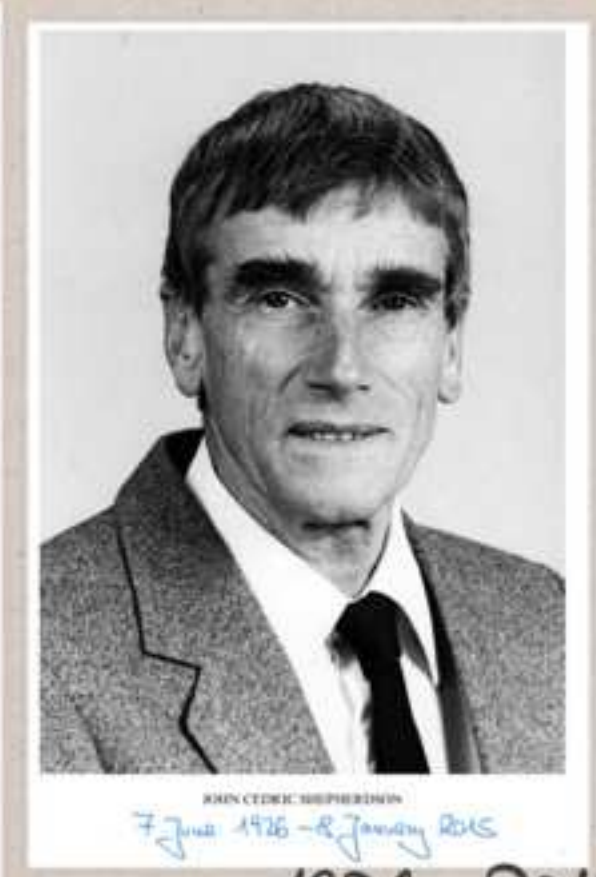
Closeup of input/output and control readers

Also known as	IBM Automatic Sequence Controlled Calculator (ASCC)
Developer	Howard Aiken / IBM
Release date	August 7, 1944; 77 years ago
Power	5 horsepower (3.7 kW)
Dimensions	816 cubic feet (23 m^3) – 51 feet (16 m) in length, 8 feet (2.4 m) in height, and 2 feet (0.61 m) deep
Mass	9,445 pounds (4.7 short tons; 4.3 t)
Successor	Harvard Mark II

Marvin Minsky

Minsky in 2008

Born	Marvin Lee Minsky August 9, 1927 New York City, New York, U.S.
Died	January 24, 2016 (aged 88) Boston, Massachusetts, U.S.
Nationality	American
Citizenship	United States



JOHN CEDRIC SHEPHERDSON

7 June 1926 – 8 January 2015

1926 – 2015

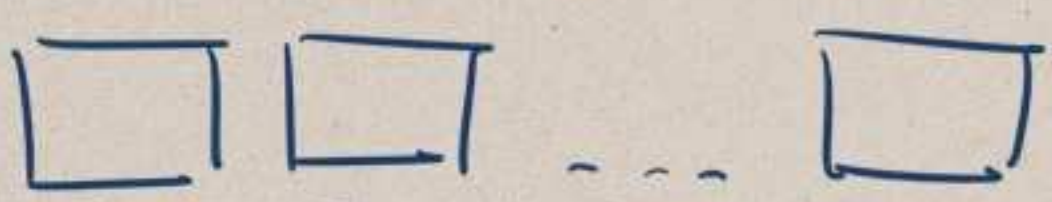
Howard E. Storgis

Zdzislaw A. Melzak

(1926 –)

University of British Columbia

A register machine has a finite number of states and a finite number of registers that can store information (i.e., elements of Σ^*).



n units of storage

Two types of INSTRUCTIONS

Type I $+(k, \sigma, s)$ $\sigma \in \Sigma$, $s \in S$, $k \in \mathbb{N}$

This instruction reads w in R_k adds σ to it: R_k now contains $w\sigma$; and moves to state s.

Type II $-(k, s, s')$ $k \in \mathbb{N}$, $s, s' \in S$.

This instruction reads w in R_k; checks whether $w = \varepsilon$.

If $w = \varepsilon$, then moves to state s'.

If $w \neq \varepsilon$, so $w = w'\sigma$. Then change R_k to w' and move to state s.

Let us write I for the set of instructions.

A RM program is a function from $S \longrightarrow I$.

If P is a program, only finitely many $k \in \mathbb{N}$ show up in ran(P). The maximum of this set is called the upper register index of P.

INTERPRET REGISTER MACHINES AS MODELS OF COMPUTATION

Again define

$$\Sigma := \Sigma \cup \{R\} \cup \{\sigma_s ; s \in S\}$$

↑ STATE SYMBOLS

Then words in Σ^* with a unique state symbol at the beginning of the word are RM configurations

$\sigma_s R w_0 R w_1 R w_2 \ldots R w_n$

finitely many occurrence of the symbol R

interpreted as

the machine has w_i in register R_i and is in state s.

Define the transition function Φ on words of Σ^*:

$$w = \sigma_s R w_0 R w_1 \ldots R w_n$$

RESULTING STATE

$\Phi(P, w)$:

t
t'
t

if $P(s) = +(k, \sigma, t)$ then

$\sigma_t R w_0 \ldots R w_{k-1} R w_k \sigma R w_{k+1} \ldots R w_n$

if $P(s) = -(k, t, t')$ and $w_k = \varepsilon$

$\sigma_{t'} R w_0 \ldots R w_{k-1} R \varepsilon R w_{k+1} \ldots R w_n$

if $P(s) = -(k, t, t')$ and $w_k = w'\sigma$

$\sigma_t R w_0 \ldots R w_{k-1} R w' R w_{k+1} \ldots R w_n$.

and the halting bit of $\Phi(P, w)$ is set to 1 iff the resulting state (t or t') is $= s_1$.

Lecture V: 23 NOVEMBER 16–18